МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
На засіданні кафедри
економічної кібернетики і
системного аналізу
Протокол № 1 від 22.08.2023 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

_____ Каріна НЕМАШКАЛО

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ
робоча програма навчальної дисципліни (РПНД)

Галузь знань	**12 «Інформаційні технології»**
Спеціальність	**124 «Системний аналіз»**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**Управління складними системами**

Статус дисципліни	**вибіркова**
Мова викладання, навчання та оцінювання	**українська**

Розробники:
к.е.н., доцент _____ Роман ЯЦЕНКО

викладач _____ Антон ЯКОВЛЄВ

Завідувач кафедри
економічної кібернетики
і системного аналізу _____ Лідія ГУР'ЯНОВА

Гарант програми _____ Оксана ПАНАСЕНКО

Харків
2024

ВСТУП

З уведенням нових технологій та розширенням обсягу задач, які вирішуються програмними засобами, важливо освоювати не лише традиційні, але і сучасні підходи до програмування. Сучасні парадигми, такі як функціональне програмування, об'єктно-орієнтоване програмування, паралельне програмування та інші, надають здобувачам широкий арсенал інструментів для ефективного та гнучкого розв'язання завдань у сфері системного аналізу. Вивчення сучасних технологій програмування сприяє розвитку глибокого розуміння концепцій розробки програмного забезпечення, збільшує конкурентоспроможність випускників на ринку праці. Крім того, ця дисципліна сприяє формуванню критичного мислення, необхідного для швидкого адаптування до вимог індустрії програмного забезпечення, що постійно змінюється.

Мета навчальної дисципліни – ознайомлення здобувачів із передовими технологіями програмування, розвиток їхніх аналітичних та творчих навичок для ефективного використання різноманітних підходів у сфері розробки програмного забезпечення. У якості головного інструменту обрано сучасну мову програмування Python.

Основним завданням вивчення навчальної дисципліни "Сучасні технології програмування" є формування у здобувачів глибокого розуміння та практичних навичок застосування сучасних підходів до програмування, що дозволить їм ефективно вирішувати завдання системного аналізу та розробки програмного забезпечення.

Предметом вивчення навчальної дисципліни "Сучасні технології програмування" є аналіз та вивчення передових технологій та парадигм програмування, таких як функціональне програмування, об'єктно-орієнтоване програмування, паралельне програмування. Здобувачі освоюють концепції та практичні аспекти цих парадигм, набуваючи навичок для використання різноманітних підходів у сучасній розробці програмного забезпечення з урахуванням потреб системного аналізу.

Об'єктом навчальної дисципліни "Сучасні технології програмування" для спеціальності «Системний аналіз» є теоретичні та практичні аспекти передових технологій та парадигм програмування. Здобувачі вивчають концепції, мови та інструменти, спрямовані на оптимізацію розробки програмного забезпечення, що є важливим для їхньої компетентності у сфері системного аналізу.

Результати навчання та компетентності, які формує навчальна дисципліна визначено в табл. 1.

Таблиця 1

Результати навчання та компетентності, які формує навчальна дисципліна

Результати навчання	**Компетентності, якими повинен оволодіти здобувач освіти**
РН 6. Знати та вміти застосовувати основні методи постановки та вирішення	КЗ 1. Здатність до абстрактного мислення, аналізу та синтезу

задач системного аналізу в умовах невизначеності цілей, зовнішніх умов та конфліктів.	
РН 8. Володіти сучасними методами розробки програм і програмних комплексів та прийняття оптимальних рішень щодо складу програмного забезпечення, алгоритмів процедур і операцій.	КФ 6. Здатність до комп'ютерної реалізації математичних моделей реальних систем і процесів; проектувати, застосовувати і супроводжувати програмні засоби моделювання, прийняття рішень, оптимізації, обробки інформації, інтелектуального аналізу даних КФ 8. Здатність організовувати роботу з аналізу та проектування складних систем, створення відповідних інформаційних технологій та програмного забезпечення
РН 9. Вміти створювати ефективні алгоритми для обчислювальних задач системного аналізу та систем підтримки прийняття рішень.	КФ 8. Здатність організовувати роботу з аналізу та проектування складних систем, створення відповідних інформаційних технологій та програмного забезпечення
РН 13. Проектувати, реалізовувати, тестувати, впроваджувати, супроводжувати, експлуатувати програмні засоби роботи з даними і знаннями в комп'ютерних системах і мережах	КФ 1. Здатність використовувати системний аналіз як сучасну міждисциплінарну методологію, що базується на прикладних математичних методах та сучасних інформаційних технологіях і орієнтована на вирішення задач аналізу і синтезу технічних, економічних, соціальних, екологічних та інших складних систем КФ 7. Здатність використовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей та прогнозування поведінки конкретних систем а саме: об'єктно-орієнтований підхід при проектуванні складних систем різної природи, прикладні математичні пакети, застосування баз даних і знань

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Зміст навчальної дисципліни

Змістовий модуль 1. Парадигми програмування на Python

Тема 1. Еволюція парадигм програмування

1.1. Визначення термінів та важливість розуміння еволюції парадигм програмування.

Вступ до ключових понять, таких як "парадигма програмування" та обґрунтування актуальності вивчення їхньої еволюції для сучасних розробників.

1.2. Процедурне програмування: Основи та структури.

Розгляд основ процедурного програмування, важливість структурованості та використання підпрограм для створення читабельного коду. Аналіз впливу цієї парадигми на розвиток програмування.

1.3. Об'єктно-орієнтоване програмування: Принципи та практика.

Детальний огляд основ ООП, зосереджений на об'єктах, класах, спадкуванні та інших ключових концепціях. Розгляд впливу ООП на дизайн програмного забезпечення.

1.4. Функціональне програмування: Підходи та переваги.

Вивчення ідеї функціонального програмування, базових принципів та його переваг. Аналіз впливу на підходи до розробки та структуру коду.

1.5. Сучасні тенденції та перспективи: Від паралельного програмування до веб-розробки.

Розгляд сучасних напрямків у програмуванні, таких як паралельне програмування та веб-розробка. Огляд тенденцій та перспектив розвитку в сучасному програмуванні.

Тема 2. Процедурне програмування та структури даних

2.1. Вступ до Python та основ процедурного програмування.

Введення до мови програмування Python та основних концепцій процедурного програмування, включаючи функції, змінні та керування потоком виконання.

2.2. Структури даних у Python: Списки та кортежі.

Розгляд вбудованих структур даних у Python, зокрема списків та кортежів. Освіта щодо їхнього використання та основних операцій.

2.3. Робота з рядками та словниками в Python.

Вивчення роботи з рядками та словниками в Python, включаючи операції, методи та важливі концепції для ефективної обробки текстових та асоціативних даних.

2.4. Введення до функцій у Python та їх роль у процедурному програмуванні.

Огляд концепцій функцій у Python, їх створення, параметри та повернення значень, а також їх роль у структурах процедурного програмування.

2.5. Процедурне програмування в Python: Приклади та практичні завдання.

Застосування набутих знань через розв'язання практичних завдань та аналіз прикладів реалізації процедурного програмування в Python.

Тема 3. Колекції Python

3.1. Введення в колекції та базові структури даних.

Огляд понять колекцій та базових структур даних у Python, таких як списки, кортежі, множини та словники. Визначення їхнього призначення та основних властивостей.

3.2. Робота зі списками та кортежами в Python: Операції та методи.

Детальний розгляд списків та кортежів, включаючи операції, індексацію, зрізи та корисні методи для ефективної обробки послідовностей в Python.

3.3. Множини в Python та їхнє використання.

Вивчення множин як унікальних та неупорядкованих колекцій у Python. Розгляд операцій та особливостей множин для різноманітних завдань.

3.4. Словники в Python: Робота з асоціативними даними.

Аналіз словників як ключ-значення пар для зберігання та роботи з асоціативними даними. Огляд методів та операцій для ефективної маніпуляції словниками.

3.5. Порівняння та вибір оптимальних колекцій: Критерії та приклади.

Розгляд критеріїв вибору колекцій в залежності від типу завдань та оптимального використання кожної структури даних. Практичні поради щодо вибору колекцій в різних сценаріях.

Тема 4. Рекурсивні функції

4.1. Основи рекурсії: Визначення та базові концепції.

Введення у концепцію рекурсії, пояснення визначення та базових ідей. Аналіз рекурсивних викликів та їх важливість в програмуванні.

4.2. Структура рекурсивних функцій в Python: Виклик та базовий випадок.

Розгляд структури рекурсивних функцій у Python, включаючи виклик функції та визначення базового випадку. Пояснення механізму роботи рекурсії на прикладах.

4.3. Приклади рекурсивних функцій: Факторіал та числа Фібоначчі.

Детальний розгляд конкретних прикладів рекурсивних функцій, таких як обчислення факторіалу та чисел Фібоначчі. Аналіз їхнього коду та викликів.

4.4. Рекурсія та робота зі списками: Пошук та модифікація.

Вивчення використання рекурсії для пошуку та модифікації списків в Python. Розгляд прикладів рекурсивного обходження списків.

4.5. Оптимізація та обмеження рекурсії: Мемоізація та глибина викликів.

Розгляд методів оптимізації рекурсії, таких як мемоізація для уникнення зайвих обчислень. Обговорення обмежень та глибини викликів рекурсивних функцій.

Тема 5. Функціональне програмування

5.1. Основні принципи функціонального програмування: Імутабельність та вищі порядкові функції.

Розгляд ключових принципів функціонального програмування, таких як імутабельність даних та використання вищих порядкових функцій. Аналіз їхньої ролі у створенні чистого та ефективного коду в Python.

5.2. Лямбда-функції та функції вищих порядків: Використання в Python.

Вивчення лямбда-функцій та їхнього використання, а також аналіз функцій вищих порядків, таких як map, filter та reduce. Демонстрація їхнього застосування в різних сценаріях.

5.3. Рекурсія та неявна рекурсія в функціональному стилі.

Розгляд використання рекурсії та неявної рекурсії в функціональному програмуванні. Аналіз прикладів та порівняння з іншими підходами.

5.4. Функціональні структури даних: Списки, відображення та інші.

Детальний огляд функціональних структур даних, таких як імутабельні списки та відображення. Розгляд їхнього використання та переваг в функціональному стилі.

5.5. Монади та їх роль у функціональному програмуванні в Python.

Вивчення концепції монад та їхнього використання у функціональному програмуванні в Python. Розгляд ролі монад у роботі із сторонніми ефектами та збереженні імутабельності.

Тема 6. Декоратори та замикання

6.1. Основи замикань: Визначення та приклади використання.

Введення у концепцію замикань, їхня роль у функціональному програмуванні та приклади використання для збереження стану та інкапсуляції.

6.2. Функції вищих порядків та замикання: Поєднання концепцій.

Розгляд функцій вищих порядків та їхнього впливу на замикання. Детальний аналіз використання замикань в поєднанні з функціями вищих порядків.

6.3. Основи декораторів: Визначення та синтаксис.

Огляд концепції декораторів, їх роль у підвищенні функціональності та зручний синтаксис. Пояснення того, як декоратори можна застосовувати до функцій.

6.4. Створення та використання декораторів в Python: Практичні аспекти.

Практичні приклади створення та використання декораторів у Python. Аналіз застосування для забезпечення логування, кешування та інших завдань.

6.5. Застосування замикань у декораторах: Гнучкість та розширюваність.

Розгляд можливостей застосування замикань у створенні гнучких та розширюваних декораторів. Аналіз патернів та стратегій використання замикань у декораторах.

Змістовий модуль 2. Об’єктно-орієнтоване та асинхронне програмування

Тема 7. Об’єктно-орієнтоване програмування

7.1. Основні поняття ООП: Класи та об'єкти.

Визначення основних термінів об'єктно-орієнтованого програмування, таких як класи та об'єкти. Аналіз структури класу та його ролі у створенні об'єктів.

7.2. Інкапсуляція та наслідування: Принципи та приклади в Python.

Розгляд інкапсуляції як принципу об'єктно-орієнтованого програмування та наслідування як механізму відновлення та розширення класів. Демонстрація їхнього використання в мові програмування Python.

7.3. Поліморфізм та перевантаження операторів в Python.

Вивчення концепції поліморфізму та його варіантів у Python, таких як перевантаження операторів та функцій. Пояснення, як це сприяє зручності та читабельності коду.

7.4. Декоратори та метакласи: Розширення можливостей ООП.

Розгляд застосування декораторів та метакласів для розширення можливостей об'єктно-орієнтованого програмування в Python. Практичні приклади використання для створення динамічних та гнучких класів.

7.5. Шаблони проектування та їхнє використання в Python.

Введення до шаблонів проектування та їхнього використання в розробці програм на мові Python. Аналіз популярних шаблонів та їх приклади в реальних сценаріях.

Тема 8. Перевантаження операторів

8.1. Основи перевантаження операторів: Визначення та необхідність.

Розгляд основних понять перевантаження операторів в Python, пояснення, чому це важливо для створення зручного та ефективного коду. Аналіз ролі перевантажених операторів у роботі з об'єктами.

8.2. Перевантаження арифметичних операторів: +, -, *, / та інші.

Детальний огляд перевантаження арифметичних операторів, таких як додавання, віднімання, множення та ділення. Пояснення процесу визначення поведінки цих операторів для користувацьких об'єктів.

8.3. Перевантаження порівняння та логічних операторів: <, >, ==, and, or і т.д.

Розгляд перевантаження операторів порівняння та логічних операторів. Пояснення, як забезпечити коректне порівняння та визначення умов для користувацьких класів.

8.4. Правила перевантаження індексації та зрізів: [ ] та [:].

Вивчення перевантаження операторів індексації та зрізів для користувацьких об'єктів. Пояснення, як створювати об'єкти, які можна індексувати та зрізати.

8.5. Створення власних перевантажених операторів: Принципи та практика.

Огляд можливостей створення власних перевантажених операторів в Python. Практичні поради щодо використання цього механізму для покращення читабельності та логіки коду.

Тема 9. Обробка виняткових ситуацій

9.1.Введення в обробку виняткових ситуацій: Поняття та значення.

Визначення основних термінів та концепцій обробки виняткових ситуацій в Python, пояснення, чому це важливо для забезпечення надійності програмного забезпечення. Розгляд ролі винятків у виявленні та обробці помилок.

9.2. Вбудовані винятки та їхнє використання: Приклади та сценарії.

Детальний огляд вбудованих винятків у Python, їхніх характеристик та практичне використання в програмуванні. Аналіз сценаріїв, де винятки є корисними для виявлення та обробки помилок.

9.3. Блоки try-except та їхнє використання: Захист від помилок.

Розгляд конструкції блоків try та except для обробки виняткових ситуацій в Python. Пояснення, як за допомогою цих блоків можна забезпечити гнучку та контрольовану взаємодію з помилками.

9.4. Ключові слова else та finally: Додаткові можливості обробки виняткових ситуацій.

Вивчення використання ключового слова else для визначення коду, який виконується при відсутності винятків, та ключового слова finally для коду, що

гарантовано виконується навіть після виникнення винятку. Пояснення їх ролі та можливостей.

9.5. Створення та власні винятки: Керування власними винятковими ситуаціями.

Огляд процесу створення власних виняткових класів для більш ефективного та контрольованого управління винятками. Практичні приклади використання власних винятків для покращення структури коду та передачі корисної інформації про помилки.

Тема 10. Асинхронне програмування

10.1.Введення в асинхронне програмування: Поняття та переваги.

Розгляд основних термінів та концепцій асинхронного програмування в Python, включаючи корутини, задачі та інші елементи. Пояснення переваг такого підходу, зокрема покращення ефективності використання ресурсів та здатність до паралельної обробки задач.

10.2. Корутини та ключові слова async та await: Основи асинхронного коду.

Детальний огляд корутин, а також введення ключових слів async та await у Python для визначення асинхронних функцій та їхнього виклику. Пояснення, як ці інструменти дозволяють створювати ефективний асинхронний код.

10.3. Модуль asyncio та його функціонал: Планування та виконання завдань.

Вивчення модуля asyncio як основного інструмента для реалізації асинхронного програмування в Python. Аналіз його функціоналу, включаючи планування та виконання асинхронних задач.

10.4. Обробка асинхронних винятків та подій: Забезпечення стабільності.

Розгляд методів обробки винятків та подій в асинхронному коді, зокрема використання конструкцій try-except для коректної взаємодії з асинхронними функціями та винятками.

10.5. Асинхронні бібліотеки та патерни: Використання у реальних проектах.

Вивчення популярних асинхронних бібліотек та патернів для реалізації асинхронного програмування в реальних проектах. Практичні приклади використання та поради щодо ефективного розроблення асинхронних додатків в Python.

Перелік лабораторних занять за навчальною дисципліною наведено в табл. 2.

Таблиця 2

Перелік лабораторних занять

Назва теми	Зміст
Тема 1.	Лабораторна робота "Історія та розвиток парадигм програмування" • Вивчення та порівняння основних парадигм програмування. • Аналіз історії еволюції мов програмування та їхніх впливів.

Тема 2.	Лабораторна робота "Основи процедурного програмування в Python" • Розробка програм з використанням процедурного підходу. • Робота зі структурами даних та їх використання в процедурному контексті.
Тема 3.	Лабораторна робота "Робота з колекціями у Python" • Використання вбудованих колекцій для розв'язання задач. • Реалізація власних класів та структур даних.
Тема 4.	Лабораторна робота "Рекурсивні функції в Python" • Створення та оптимізація рекурсивних функцій. • Вирішення завдань, що вимагають використання рекурсії.
Тема 5.	Лабораторна робота "Впровадження функціонального програмування в Python" • Реалізація функціональних конструкцій у Python. • Робота з вищим порядком функцій та обробка спискових даних.
Тема 6.	Лабораторна робота "Застосування декораторів та замикань" • Створення та використання декораторів для модифікації функціональності. • Робота з замиканнями для зберігання стану.
Тема 7.	Лабораторна робота "Розробка об’єктно-орієнтованих програм в Python" • Створення класів та об'єктів у Python. • Реалізація основних принципів ООП: інкапсуляція, спадкування, поліморфізм.
Тема 8.	Лабораторна робота "Перезавантаження операторів в Python" • Впровадження перевантаження операторів для користувацьких класів. • Робота з різними типами операторів.
Тема 9.	Лабораторна робота "Обробка виняткових ситуацій в Python" • Використання блоків try, except, else та finally. • Реалізація власних виняткових ситуацій та їх обробка.
Тема 10.	Лабораторна робота "Асинхронне програмування в Python" • Створення асинхронних функцій та корутин. • Використання бібліотеки asyncio для організації асинхронних задач.

Перелік самостійної роботи за навчальною дисципліною наведено в табл. 3.

Таблиця 3

Перелік самостійної роботи

Назва теми	**Зміст**
Тема 1.	Провести аналіз історії та еволюції однієї з парадигм програмування (наприклад, об'єктно-орієнтованої) та підготувати короткий огляд.
Тема 2.	Створити програму, використовуючи процедурний стиль програмування та вбудовані структури даних.
Тема 3.	Реалізувати алгоритм, використовуючи різні вбудовані колекції Python, та порівняти їхню ефективність.

Тема 4.	Реалізувати рекурсивні алгоритми для задач, які допускають рекурсивний підхід.
Тема 5.	Створити простий програмний код, використовуючи функціональний підхід, для вирішення задачі, яку раніше було вирішено імперативним способом.
Тема 6.	Створити власний декоратор для вимірювання часу виконання функцій.
Тема 7.	Розробити систему класів для предметної області, використовуючи принципи ООП (наслідування, інкапсуляцію, поліморфізм).
Тема 8.	Перевантажити оператори для користувацького класу та використати їх у певних сценаріях.
Тема 9.	Реалізувати код, який обробляє винятки та повертає коректні повідомлення про помилки.
Тема 10.	Реалізувати асинхронний код для вирішення задачі, яка потребує паралельного виконання багатьох операцій.

Кількість годин лекційних і лабораторних занять та годин самостійної роботи наведено в робочому плані (технологічній карті) з навчальної дисципліни.

МЕТОДИ НАВЧАННЯ

При викладанні навчальної дисципліни “Сучасні технології програмування” для активізації навчального процесу передбачено застосування сучасних навчальних технологій, таких, як: проблемні лекції; міні-лекції; презентації, виконання індивідуальних творчих завдань.

Проблемна лекція «Виклики та переваги: адаптація до сучасних парадигм програмування» в темі 1 спрямована на розвиток логічного мислення здобувачів. Коло питань теми обмежується двома-трьома ключовими моментами, увага здобувачів концентрується на матеріалі, що не знайшов відображення в підручниках, використовується досвід закордонних навчальних закладів з роздачею здобувачам під час лекцій друкованого матеріалу та виділенням головних висновків з питань, що розглядаються. При викладанні лекційного матеріалу здобувачам пропонуються питання для самостійного розмірковування. При цьому лектор задає запитання які спонукають здобувача шукати розв’язання проблемної ситуації. Така система примушує здобувачів сконцентруватися і почати активно мислити в пошуках правильної відповіді.

На початку проведення проблемної лекції необхідно чітко сформулювати проблему, яку необхідно вирішити здобувачам. При викладанні лекційного матеріалу слід уникати прямої відповіді на поставлені запитання, а висвітлювати матеріал таким чином, щоб отриману інформацію здобувач міг використовувати при розв’язанні проблеми.

Міні-лекція «Монади, каррірovані функції та застосування в реальних проектах» в темі 5 передбачає викладення навчального матеріалу за короткий проміжок часу й характеризуються значною ємністю, складністю логічних побудов, доказів та узагальнень. Міні-лекції проводяться, як правило, як частина

заняття-дослідження. На початку проведення міні-лекції за вказаною вище темою лектор акцентує увагу здобувачів на необхідності представити викладений лекційний матеріал у так званому структурно-логічному вигляді. На розгляд виносяться питання, які зафіксовані у плані лекцій, але викладаються вони стисло. Лекційне заняття, проведене у такий спосіб, побуджає у здобувача активність та увагу при сприйнятті матеріалу, а також спрямовує його на використання системного підходу при відтворенні інформації, яку він одержав від викладача.

Презентації результатів виконання самостійного завдання щодо асинхронного програмування в Python за темою 10 – це виступи перед аудиторією, що використовуються для представлення звіту про виконання індивідуальних завдань. Однією з позитивних рис презентації та її переваг при використанні в навчальному процесі є обмін досвідом, який здобули здобувачі при роботі над індивідуальним завданням або у певній малій групі.

ФОРМИ ТА МЕТОДИ ОЦІНЮВАННЯ

Університет використовує 100 бальну накопичувальну систему оцінювання результатів навчання здобувачів вищої освіти.

Система оцінювання сформованих компетентностей у здобувачів враховує види занять, які згідно з програмою навчальної дисципліни передбачають лекційні, лабораторні заняття, а також виконання самостійної роботи. Контрольні заходи включають:

поточний контроль, що здійснюється протягом семестру під час проведення лекційних, лабораторних занять і оцінюється сумою набраних балів (максимальна сума – 60 балів; мінімальна сума, що дозволяє здобувачу складати іспит, – 35 балів);

модульний контроль, що проводиться у формі поточних контрольних робіт за змістові модулі та має на меті *інтегровану* оцінку результатів навчання здобувача після вивчення матеріалу з логічно завершеної частини дисципліни – змістового модуля 1 та 2.

Поточний та модульний контроль оцінювання знань протягом змістових модулів включає:

виконання самостійних індивідуальних завдань. Загальна кількість балів – 20;

презентація результатів виконання самостійного завдання щодо асинхронного програмування в Python – 20 балів;

поточні контрольні роботи – 20 балів.

Підсумковий контроль знань та компетентностей здобувачів з навчальної дисципліни здійснюється на підставі проведення семестрового екзамену, завданням якого є перевірка розуміння здобувачем програмного матеріалу в цілому, логіки та взаємозв'язків між окремими розділами, здатності творчого

використання накопичених знань, вміння формулювати своє ставлення до певної проблеми навчальної дисципліни тощо.

Максимальна сума балів, яку може отримати здобувач вищої освіти під час екзамену (іспиту) – 40 балів. Мінімальна сума, за якою екзамен (іспит) вважається складеним – 25 балів.

Екзаменаційний білет охоплює програму дисципліни і передбачає визначення рівня знань та ступеня опанування здобувачами компетентностей. Кожен екзаменаційний білет складається із стереотипного, діагностичного та евристичного завдання, які передбачають вирішення типових професійних завдань фахівця на робочому місці та дозволяють діагностувати рівень теоретичної підготовки здобувача і рівень його компетентності з навчальної дисципліни.

Більш детальну інформацію щодо системи оцінювання наведено в робочому плані (технологічній карті) з навчальної дисципліни.

Приклад екзаменаційного білету

1. Стереотипне завдання: Еволюція парадигм програмування

1.1. Розкрийте основні етапи еволюції парадигм програмування, вказавши ключові особливості кожного етапу.

1.2. Які виклики виникають перед програмістами на сучасному етапі розвитку парадигм програмування? Зазначте приклади інновацій, що впливають на програмування.

2. Діагностичне завдання: Функціональне програмування в Python

2.1. Поясніть основні концепції функціонального програмування. Які переваги та особливості цієї парадигми в порівнянні з імперативним програмуванням?

2.2. Наведіть приклади використання функціональних елементів в мові програмування Python. Зазначте, як вони сприяють чистоті та зрозумілості коду.

3. Евристичне завдання: Об'єктно-орієнтоване програмування

3.1. Поясніть базові принципи об'єктно-орієнтованого програмування (ООП). Як вони взаємодіють між собою та сприяють вирішенню завдань програмування?

3.2. Наведіть приклад створення класу та об'єктів в мові Python. Як можна використовувати спадкування та інкапсуляцію для покращення організації коду?

Примітка: Відповіді на кожне завдання повинні бути аргументовані та містити конкретні приклади.

Критерії оцінювання

Загальна оцінка формується за наступною формулою:

кількість балів, отриманих за виконання стереотипного завдання +

кількість балів, отриманих за виконання діагностичного завдання +

кількість балів, отриманих за виконання евристичного завдання

Критерії перевірки для **стереотипного** екзаменаційного завдання (максимум – 10 балів):

1. Розуміння парадигм програмування: (до 4 балів)

Розкриття основних парадигм програмування та їх взаємодії.

2. Опис основних концепцій: (до 4 балів)

Чітке пояснення концепцій, характерних для кожної з парадигм.

3. Порівняння парадигм: (до 2 балів)

Висновки щодо переваг та недоліків різних парадигм та їх практичного застосування.

Критерії перевірки для **діагностичного** екзаменаційного завдання (максимум – 15 балів):

1. Розуміння асинхронності в Python: (до 5 балів)

Визначення основних принципів асинхронного програмування та їхнє впровадження в Python.

2. Використання асинхронних інструментів: (до 7 балів)

Демонстрація навичок використання асинхронних бібліотек та інструментів у Python.

3. Оптимізація роботи з асинхронністю: (до 3 балів)

Пояснення методів оптимізації роботи з асинхронністю та їх впровадження в конкретних сценаріях.

Критерії перевірки для **евристичного** екзаменаційного завдання (максимум – 15 балів):

1. Розуміння ООП: (до 5 балів)

Чітке визначення основних принципів об'єктно-орієнтованого програмування.

2. Створення класів та об'єктів: (до 7 балів)

Наведення прикладів створення класів та роботи з об'єктами в Python.

3. Наслідування та поліморфізм: (до 3 балів)

Детальний опис використання наслідування та поліморфізму в контексті мови програмування Python.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Васильєв О.М. Програмування мовою Python. – Л.: Bohdan Books, 2022. – 504 с.
2. Беррі П. Head First. Python. – Х.: ФАБУЛА, 2021. – 624 с.
3. Маттес Е. Пришвидшений курс Python. – Л.: Видавництво Старого Лева, 2021. – 600 с.
4. Щербаков, О. В. Основи об'єктно-орієнтованого програмування [Електронний ресурс] : навч. посіб. / О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко ; Харківський національний економічний університет ім. С. Кузнеця. - Електрон. текстові дан. (2,13 МБ). - Харків : ХНЕУ ім. С. Кузнеця, 2019. - 236 с. : іл. - Загол. з титул. екрану. - Бібліогр.: с. 231-232.- Режим доступу: http://www.repository.hneu.edu.ua/handle/123456789/23847
5. Щербаков О. В. Сучасні тенденції розвитку мов програмування на прикладі Java та C# / О.В. Щербаков, Ю.І. Скорін // Інформаційні технології та системи : матеріали міжнар. наук.-практ. конф., 8 – 9 квіт. 2021 р. : тези допов. – Харків : ХНЕУ ім. С. Кузнеця, 2021. – С. 36. - Режим доступу: http://www.repository.hneu.edu.ua/handle/123456789/25604

Додаткова

6. Фрімен Е., Робсон Е. Head First. Патерни проєктування. – Х.: ФАБУЛА, 2020. – 672 с.
7. Мартін Р. Чистий кодер. – Х.: ФАБУЛА, 2023. – 256 с.
8. Креневич А.П. Python у прикладах і задачах. Частина 1. Структурне програмування Навчальний посібник із дисципліни "Інформатика та програмування" – К.: ВПЦ "Київський Університет", 2017. – 206 с.

9. Креневич А.П. Python у прикладах і задачах. Частина 2. Об’єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 с.

10. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки", спеціалізації "Інформаційні технології в біології та медицині" / А. В. Яковенко. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 195 с.

Інформаційні ресурси

11. Патерни проектування [Електронний ресурс]. – Режим доступа : https://refactoring.guru/uk/design-patterns.

12. W3Schools Online Web Tutorials Distribution [Електронний ресурс]. – Режим доступа : https://www.w3schools.com/.

13. Heroku: Cloud Application Platform [Електронний ресурс]. – Режим доступа : https://www.heroku.com/.

14. Сайт Національної бібліотеки України ім. Вернадського [Електронний ресурс]. – Режим доступа : www.nbuv.gov.ua.

15. Сайт Python Programming Language [Електронний ресурс]. – Режим доступа : https://www.python.org/.

16. Teach Python 3 and web design with 200+ exercises [Електронний ресурс]. – Режим доступа : https://snakify.org/en/